

# Lec 1

## Compilers

- What is Compilers?
- Phases of Compilers?
- Function of Phases
- The Programs we use to Process it.

→ our goals.

Translators → do the same Function of Compilers.

- Types of Translators? ~~and~~ their Functions?

### Preprocessor:-

• micro inst يأخذ

### in Programming

#include ..... السطر ده بتاع  
(Preprocessor)

→ (translation) يحول ~~من لغة~~ من لغة ~~الى لغة~~ الى لغة

(another high level language) الى (high level language)

- Some Compilers can give us executable Code (ready) or assembly Code (need some operations to be ready)

→ Translator has to get two programs do the same function to translate between them.

## Types of Translator

### → Pre Processor

(level language) to (level language) ←

### → Assembler

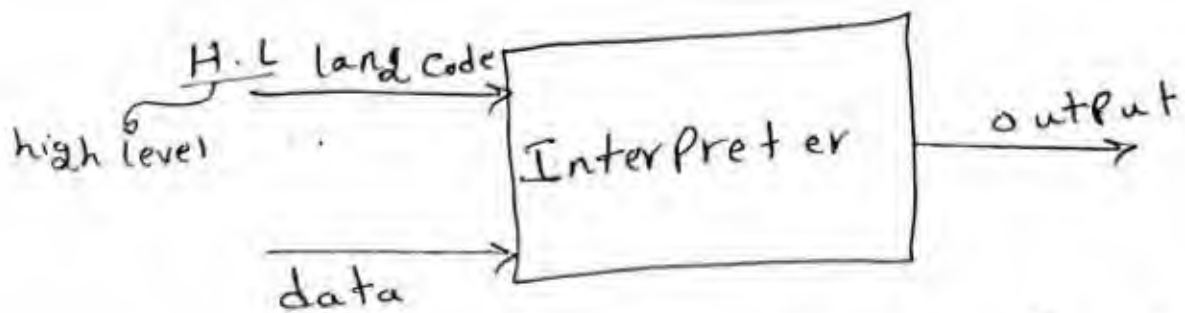
→ It is one to one →

(one instruction) ← (assembly inst.) ←

(Disassembler) ← ← (assembler)

← (Assembler) ← ← (machine) ← ← (run)

← ← (machine) ← ← (run)



→ difference between Compiler & interpreter

↳ on pdf slides.

← (interpreter) أسرع في وقت (run)

← (in total) ال (Compiler) أسرع.

← ال (interpreter) قابل (error) يؤثر على خروجه  
 بينما يتوقف تماماً إذا ال (Compiler) فيشكل العمل.

### Compilers

→ classified for instruction

a) single Pass.

b) multiple Pass.

→ classified with consideration of optimization:

a) global

b) local

→ There are others do the same function or similar function to the compilers.

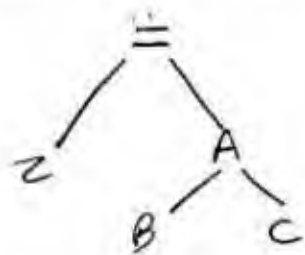
Like cross-Compiler, source-to-source.

→ Preprocessor is similar to source-to-source in the definition.

→ difference between source-to-source Compiler and Preprocessor ⇒ report.

### Architecture of Compilers

→ Front-end → away from machine  
→ machine independent.



(ADD(A, B, C))

→ reintermediate representation  
→ not related to machine.

→ Analysis goes in three parts.

Analyses → lexical (scanner)  
→ Syntax  
→ Semantic

Front end  $\rightarrow$  Analysis  
 $\rightarrow$  intermediate code representation.

← القياس ما بين (Front end) (back-end)  
هو علاقته بال (machine).

back-end  $\rightarrow$  if we work with local optimization.

$\rightarrow$  What is Parsing ??

$\rightarrow$  syntax-analysis  $\Rightarrow$  it's 2nd name is Parsing.

$\rightarrow$  machine independent  $\rightarrow$  global optimiser.

$\rightarrow$  machine dependent  $\rightarrow$  local " .

$\Rightarrow$  Phases we will study more:-

\* lexical

\* syntax

\* Code Generator

\* machine dependent code optimiser.

## → Compiler Function

- transform from high level lang. to low level lang.
- handling ~~of~~ for error.
- build symbol table.

## \* Symbol table

→ أى (Variable) → يحزن في (Symbol table) ويرجى مكان في الذاكرة ثم يتم تخزين ال (Symbol table) في الذاكرة أى أنه يأخذ مكانه في ال (memory).

→ search for it.

## → Lexical (scanner)

ال tokens هنا يعنى ال (word) لكن في البرنامج فقط.

→ takes statement by statement.

(Lexical analysis) قسوى ال (Command) فإمعنا.

(Preprocessor) مش بيشتغل.

→ lexemes must be meaningful as a language.

→ Syntax analysis

→ check source code for errors.

→ give us parse tree for atoms as o/p.

→ search of the sentence's structure.

intermediate lang → machine independent.

→ لا يمكن تحويله إلى لغة آلة  
لأنه ليس بـ PDF .

[7] Lec 1